The Programming Assignment Report Instructions
CSCE 221

1. The description of an assignment problem.

   Find a path to the exit in a maze that consists of rooms connected by doors. Each room is enumerated by consecutive integers, from 0 to n 1, and it can have no more than 4 doors oriented to North, South, East or West. The entry to the maze starts at the room 0 and ends at the room n^2-1. The maze can be modeled by an undirected graph, where a vertex corresponds to a room and an edge corresponds to a door.

   Design an algorithm to find a path from the entrance to the exit. Find the length of the path. Prompt the user to load the graph from an imput file. Use an adjacency list to organize the input data as a graph. \ Implement an algorithm that finds the entry-exit path. Implement a function that displays the path. Create a menu that includes the following options: – Print the adjacency list for a given graph. – Find the length of the entry-exit path generated by your algorithm. – Print out all vertices (room numbers) on the entry-exit path. Provide at least two test cases, each consisting of a maze data, an input file and an output file in text format, to verify the correctness of your program.

2. The description of data structures and algorithms used to solve the problem.

   (a) Provide definitions of data structures by using Abstract Data Types (ADTs)

   I created a struct called mazeNode and a struct called mazeVec which containes a vector of maze nodes as well as an insert maze node function. mazeNodes keep store their positive bools for north south east and west as well as list that contain the pos of adjacent nodes. Their constructor takes the bool for north, south, east, and west as well as a position value. It inserts to the adjacency list based on the position value and the bools. Maze node also supports and overloaded operator to output an adjacency list.

   (b) Write about the ADTs implementation in C++.

   mazeVec which containes a vector of maze nodes as well as an insert maze node function. The default constructor for mazeVec does not do assignment and there is no other constructor mazeNodes keep store their positive bools for north south east and west as well as list that contain the pos of adjacent nodes. Their constructor takes the bool for north, south, east, and west as well as a position value. It inserts to the adjacency list based on the position value and the bools. Maze node also supports and overloaded operator to output an adjacency list. It simply iterates through the list and outputs each value.

   (c) Describe algorithms used to solve the problem.

   The shortest path algorithm is a breadth first search. It shortest path algorithm takes a mazeVec and returns a vector with the shortest path to traverse the maze. The algorithm creates a boolean visitied array the size of the maze and initializes every value to false. Creates a parent array the size of the maze. It creates a q and pushes the first node onto the queue. It enters a while loop that continues until the exit is visited. It the creates a vector called path and backtracks from the exit node. Finding the parent of each node on the path and adding that node to path vector. The path vector will now be in backwards order so it reverses it and returns it.

   (d) Analyze the algorithms according to assignment requirements.

   This algorithm is O(n +m) where n is the number of vertices and m is the number of edges. In because in the worst case scenario every verex and every edge will be explored.

3. A C++ organization and implementation of the problem solution

   (a) Provide a list and description of classes or interfaces used by a program such as classes used to implement the data structures or exceptions.

       I created a struct called mazeNode and a struct called mazeVec which containes a vector of maze nodes as well as an insert maze node function. mazeNodes keep store their positive bools for north south east and west as well as list that contain the pos of adjacent nodes. Their constructor takes the bool for north, south, east, and west as well as a position value. It inserts to the adjacency list based on the position value and the bools. Maze node also supports and overloaded operator to output an adjacency list.

   (b) Include in the report the class declarations from a header file (.h) and their implementation from a source file (.cpp).

```
//
#include <vector>
#include <list>
using namespace std;
struct mazeNode{
    mazeNode(bool north, bool east, bool south, bool west, int p);
    int pos;
    bool n, s, e, w;
    std::list<int> adj;
    friend ostream& operator << (ostream &out, const mazeNode &node);
};

ostream& operator << (ostream &out, const mazeNode &node)
{
    list<int>::const_iterator i;
    out << node.pos;
    for (i = node.adj.begin(); i!= node.adj.end(); ++i)
        out << " -> " << *i;
    return out;
}

mazeNode::mazeNode(bool north, bool east, bool south, bool west, int p)
n(north), pos(p), e(east), s(south), w(west)
{
    if (north)
        adj.push_back(pos-4);
    if (east)
        adj.push_back(pos+1);
    if (south)
        adj.push_back(pos+4);
    if (west)
        adj.push_back(pos-1);
}
struct mazeVec{
    std::vector<mazeNode> maze;
    //std::vector<mazeNode> maze;
    //construct nodes with nsew
    //assign nodes pointers
    void insert(bool n, bool e, bool s, bool w, int p);
};

void mazeVec::insert(bool n, bool e, bool s, bool w, int p)
{
    mazeNode temp(n,e,s,w,p);
    maze.push_back(temp);
}
```

4. A user guide description how to navigate your program with the instructions how to:

   (a) compile the program: specify the directory and file names, etc.

       Compile the program by typing make.

   (b) run the program: specify the name of an executable file.

       Run the program by typing ./main. The name of the executable is main. There are 2 different sample mazes included for testing called maze1.txt and maze2.txt.

1. Specifications and description of input and output formats and files

   (a) The type of files: keyboard, text files, etc (if applicable).

   Input and output files should be text files in the maze format specified in the assignment.

   (b) Discuss possible cases when your program could crash because of incorrect input (a wrong file name, strings instead of a number, or such cases when the program expects 10 items to read and it finds only 9.)

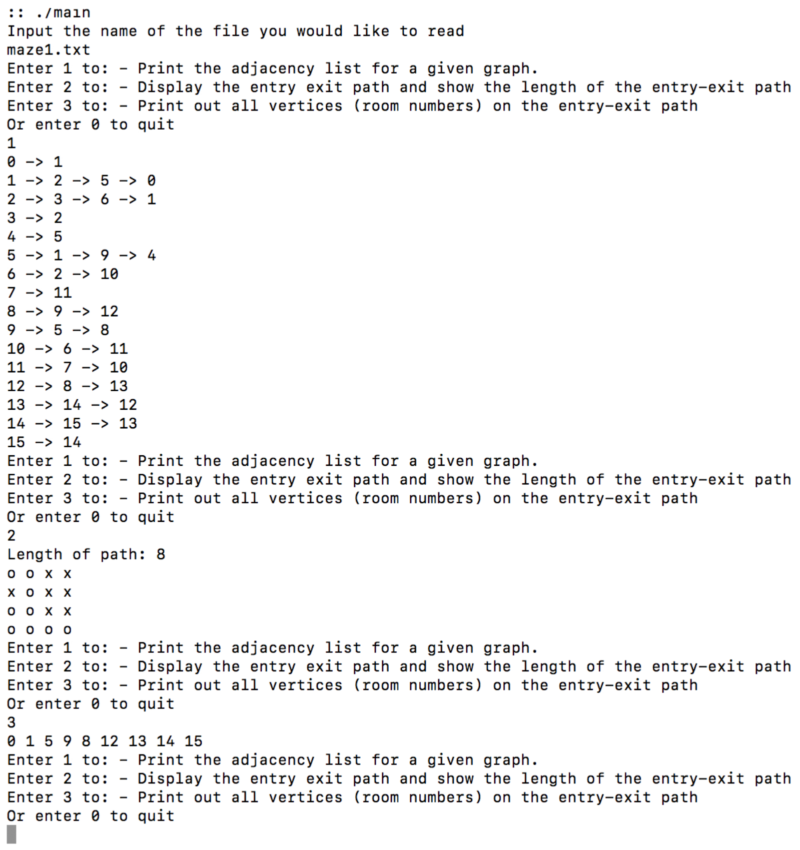
   Inputting a non existent file name or inputting a file that doesn't contain a maze.

2. Provide types of exceptions and their purpose in your program.

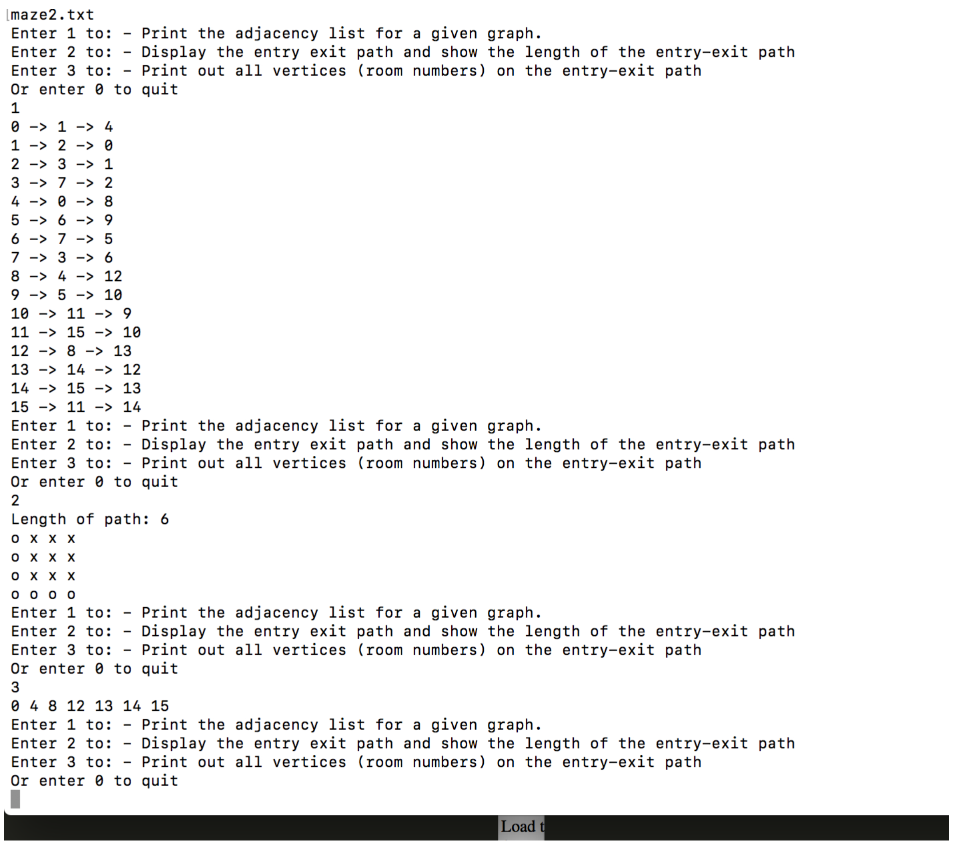
   The shortest path function throws an exception if a path to the exit cannot be found.

3. Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output.

   I did testing with the files maze1.txt and maze2.txt. I also outputted maze1out.txt and maze2out.txt.

   (a) 

```
:: ./main
Input the name of the file you would like to read
maze1.txt
Enter 1 to: - Print the adjacency list for a given graph.
Enter 2 to: - Display the entry exit path and show the length of the entry-exit path
Enter 3 to: - Print out all vertices (room numbers) on the entry-exit path
Or enter 0 to quit
1
0 -> 1
1 -> 2 -> 5 -> 0
2 -> 3 -> 6 -> 1
3 -> 2
4 -> 5
5 -> 1 -> 9 -> 4
6 -> 2 -> 10
7 -> 11
8 -> 9 -> 12
9 -> 5 -> 8
10 -> 6 -> 11
11 -> 7 -> 10
12 -> 8 -> 13
13 -> 14 -> 12
14 -> 15 -> 13
15 -> 14
Enter 1 to: - Print the adjacency list for a given graph.
Enter 2 to: - Display the entry exit path and show the length of the entry-exit path
Enter 3 to: - Print out all vertices (room numbers) on the entry-exit path
Or enter 0 to quit
2
Length of path: 8
o o x x
x o x x
o o x x
o o o o
Enter 1 to: - Print the adjacency list for a given graph.
Enter 2 to: - Display the entry exit path and show the length of the entry-exit path
Enter 3 to: - Print out all vertices (room numbers) on the entry-exit path
Or enter 0 to quit
3
0 1 5 9 8 12 13 14 15
Enter 1 to: - Print the adjacency list for a given graph.
Enter 2 to: - Display the entry exit path and show the length of the entry-exit path
Enter 3 to: - Print out all vertices (room numbers) on the entry-exit path
Or enter 0 to quit
```



```
maze2.txt
Enter 1 to: - Print the adjacency list for a given graph.
Enter 2 to: - Display the entry exit path and show the length of the entry-exit path
Enter 3 to: - Print out all vertices (room numbers) on the entry-exit path
Or enter 0 to quit
1
0 -> 1 -> 4
1 -> 2 -> 0
2 -> 3 -> 1
3 -> 7 -> 2
4 -> 0 -> 8
5 -> 6 -> 9
6 -> 7 -> 5
7 -> 3 -> 6
8 -> 4 -> 12
9 -> 5 -> 10
10 -> 11 -> 9
11 -> 15 -> 10
12 -> 8 -> 13
13 -> 14 -> 12
14 -> 15 -> 13
15 -> 11 -> 14
Enter 1 to: - Print the adjacency list for a given graph.
Enter 2 to: - Display the entry exit path and show the length of the entry-exit path
Enter 3 to: - Print out all vertices (room numbers) on the entry-exit path
Or enter 0 to quit
2
Length of path: 6
o x x x
o x x x
o x x x
o o o o
Enter 1 to: - Print the adjacency list for a given graph.
Enter 2 to: - Display the entry exit path and show the length of the entry-exit path
Enter 3 to: - Print out all vertices (room numbers) on the entry-exit path
Or enter 0 to quit
3
0 4 8 12 13 14 15
Enter 1 to: - Print the adjacency list for a given graph.
Enter 2 to: - Display the entry exit path and show the length of the entry-exit path
Enter 3 to: - Print out all vertices (room numbers) on the entry-exit path
Or enter 0 to quit
```

(b)